ISSN 0956-9979

SEPTEMBER 1993

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

Editor: **Richard Ford**

Technical Editor: **Fridrik Skulason**

Consulting Editor: **Edward Wilding**, Network Security Management, UK

IN THIS ISSUE:

• **Memory-resident scanners**. The first ever comparative review of memory-resident virus detection software reveals some surprising flaws in the protection provided by this popular line of defence.

• **Creating a boot disk**. When using disk compression software, it is vital that a bootable system disk is created before disaster strikes.

• **McAfee replaced by *Sun* executive**. John McAfee is no longer Chief Executive Officer of *McAfee Associates*.

CONTENTS

EDITORIAL

New Looks, New Ideas?

When this month's *Virus Bulletin* lands on readers' desks, there should be a bit of a surprise on opening the envelope: in its fifth year, *Virus Bulletin* has changed its appearance. Not beyond recognition, one hopes, but having listened carefully to the feedback sent in by the *Virus Bulletin* readership, we believe the new look *VB* should prove a better way to get the message across.

A lot has happened since 1989, when the very first *Virus Bulletin* rolled off the press - at which moment there was only one subscriber (editing a magazine is always a personal thing, but not usually that personal!) and there were only 14 viruses known for the IBM PC. These halcyon days are now far gone, with over three thousand viruses known to researchers, but the concept and aims of *Virus Bulletin* remain the same.

"VB has been alternately threatened, cajoled, bribed and begged to change various comments."

Not only has the number of computer viruses risen, but so has the number of anti-virus products. Although the anti-virus industry is no longer seen as the new computer gold rush, several people have made a lot of money - for example, the *McAfee Associates* initial public offering raised several million dollars for the owners.

Probably the most difficult part of producing *VB* is getting the product reviews right. *VB* is one of the very few journals to give manufacturers the chance to comment on reviews before publication. Nobody in the *Virus Bulletin* offices will ever forget the 38-page fax message which came through regarding one product review. In the hundred or so reviews published since the magazine began, *VB* has been alternately threatened, cajoled, bribed and begged to change various comments. In most cases this led to re-examination of the disputed point. Legal action has been threatened on a number of occasions, but as yet, nobody has converted their sabre-rattling into action.

The problems of the users have multiplied too. Not content with simply producing viruses, the computer underground has now organised itself into a reasonably cohesive set of groups which are quite capable of attempting *en masse* to trip up the anti-virus industry. Even so, with the world still in recession, IT Managers are finding it hard to 'sell' the concept of virus detection to higher management. Unbelievably, there are still people who think that the entire computer virus threat is something dreamed up from within the industry, in an attempt to make money from a non-existent phenomenon. The accusations that viruses are written by anti-virus companies still appear, but by now they have as little credibility as accusing the pharmaceutical industry of spreading disease.

Fortunately, things are not all doom and gloom . There is increasing cooperation within the industry, and developers and users alike are realising that spattering 'Now detects 9,323.6 viruses' across the software's box has little meaning. Governments are beginning to take the problem more seriously, and the new computing laws are bearing their first crop of cases. Thanks to the Computer Misuse Act, the UK virus writing group *ARCV* has now been shut down, with several of its members awaiting trial. Interestingly, the *ARCV's* first newsletter bragged how its members would be dodging the Special Branch. Clearly they did not dodge quickly or often enough!

If so much has changed since *Virus Bulletin* was first published, how have the objectives of the magazine fared - and are the foundations upon which the magazine were built still sound? Firstly, *VB* is a source of hard facts. It is all too easy to panic at the apocalyptic stories which can be plastered over the tabloid headlines: Jersualem, Datacrime, and Michelangelo have all been heralded as the 'end of computing as we know it'. How long will it take until the next virus catches the imagination of the press? Secondly, the aim of *Virus Bulletin* is to educate, for only by understanding the problem can computer users fight back.

Throughout all this panic and hype, *VB* has provided a rock-solid platform upon which to build the fundamentals of a good computer security policy. Though the world around has changed, the basic tenets of good security have not, and will not, for the foreseeable future. Even though the magazine looks different, the principles it is built upon have not varied. Knowledge is power. Use it!

NEWS

New CEO for McAfee

Surprise of the month in the ever-stormy anti-virus industry has been the replacement of John McAfee as Chief Execu-tive Officer of *McAfee Associates* . On August 18th, the company announced that McAfee was moving into the role of Chief Technical Officer to make way for *Sun Microsystems* executive Bill Larson. Larson, who stepped down as *SunSoft's* Vice President of Sales and Marketing, is expected to help with the strategic planning issues which will occur as the company expands and diversifies.

John McAfee, the founder of *McAfee Associates* , will assume the role of Chief Technical Officer, while remaining chairman of the board. The position is claimed to offer McAfee a greater 'hands-on and strategic involvement in the development and acquisition of new products and technologies necessary for the company's diversification'.

This change is not the only alteration to the *McAfee* team: the company recently appointed Marc LeBrun, the former director of advanced technology at *Autodesk*, as vice president of research and development. William McKiernan remains the chief operat-ing officer.

This follows the acquisition of personal database developer, *ButtonWare Inc.*, last month - a move which indicates *McAfee Associates'* desire to diversify its interests.

Phil Talsky, of *McAfee Associates* , strongly denied the rumours that after the company's poor performance since its flotation on the stock market, McAfee had been ousted by angry sharehold-ers. 'Nothing could be further from the truth' said Talsky. 'John actively wanted to spend more time on the technical issues, and this allows him to do just that' ▮

Symantec Shopping Spree

Symantec has been shopping again... and the latest tasty morsel to be snapped up by this voracious corporate is none other than *Fifth Generation Systems.*

Coming so soon after acquiring competing anti-virus software company *Certus,* this gives *Symantec* control of several different anti-virus products, including *Norton Anti-virus, Certus NOVI* , *Untouchable* and *Search and Destroy*.

Commenting on the takeover, Gordon Eubanks Jr., *Symantec's* president and CEO said '*Fifth Generation* was the pioneer in workstation back-up, just as *Peter Norton Computing* pioneered desktop recovery and anti-virus utilities.' [*! Ed.*]

The estimated cost of the acquisition is about $43 million at the current market prices, and *Symantec* will issue approximately 2.8 million shares of its common stock for the current outstanding shares of *Fifth Generation* stock ▮

Virus Prevalence Table - July 1993

Virus	Incidents	(%) Reports
Form	17	36.2%
New Zealand 2	8	17.0%
Spanish Telecom	4	8.5%
Tequila	3	6.4%
Yankee	3	6.4%
DIR-II	2	4.3%
Cascade	1	2.1%
Flip	1	2.1%
Hi-460	1	2.1%
JoJo	1	2.1%
Maltese Ameoba	1	2.1%
NoInt	1	2.1%
Parity Boot	1	2.1%
Sibel Sheep	1	2.1%
Vacsina	1	2.1%
V-Sign	1	2.1%
Total	**47**	**100.0%**

US Treasury Vx BBS: Aftermath

According to a report in the *LAN Times* , Representative Edward Markey has announced that he plans to propose legislation this autumn to make it illegal for anyone to post or disseminate computer viruses. This follows the discovery of a virus exchange Bulletin Board system running on a machine owned by the *US Treasury Department* - a system which has now been shut down.

Markey, who is chairman of the House Subcommittee on Telecommunications and Finance, which oversees computer and network security, has spoken out against computer criminals a number of times.

'The feeling among the hacker community is: if an activity is not explicitly illegal, than it is okay', said Colin Crowell, a policy analyst in Markey's office. 'The hacker community feels that we're trampling on their civil liberties, but we don't see it that way. As computer and network usage becomes more prevalent, we're seeing an explosive growth in computer and telecommuni-cations crimes, such as hacking into PBX systems.'

The BBS system in question has ceased to operate, but it would appear that the issues raised by its existence will not die away so easily. Crowell promised that the government would act, saying 'Laws are passed to deal with the 5% of the user population that engages in illegal activities, and we intend to deal with these problems with legislation.' ▮

Perry: Virus, Trojan or Joke?

It has been a confusing month for those who use both *VSUM* and *CPAV*, as *Central Point's* product has been reporting that a part of the *VSUM* package (Patricia Hoffman's hypertext virus database) is infected with the 'Perry virus'. The problem should have ended with either one side or the other backing down: after all, surely a program is either infected or it is clean. Unfortunately, things did not work out that simply.

Roots

In order to understand the relevant issues, it is necessary to turn back the clock and look at where the Perry program came from, and what it was designed to do.

Perry was written by *Interpath*, an early incarnation of John McAfee's *McAfee Associates* . The idea behind the program was a reasonable one: it would allow shareware authors to stop their product operating after a specific date. This was achieved by 'trojanising' the shareware executable with the Perry code, which would cause the executable to delete itself after a certain date.

Unfortunately, such a utility was inevitably misused. The office prankster could have a field day, 'Perry-ising' other users' software. In addition to this, the 'virus' gradually made its way into virus collections which were used for testing purposes, and the Perry trojan gradually came to be seen as a piece of malicious code. *Central Point* obtained a sample of the Perry program from the *NCSA's* virus library. Here things would (and should) have ended had it not been for an ironic twist of fate.

Patricia Hoffman's *VSUM* package contains a very early copy of *McAfee Associates* program Validate, which is used to check the integrity of the *VSUM* files. This copy of Validate was written by the same author as Perry, and just like files produced by Perry, would delete itself (in this case sometime after the year 2000). Therefore, when *Central Point* expanded its library to cover Perry, *CPAV* began reporting that the file VALIDATE.COM was infected.

Blame and Guilt

Nobody appears to be wholly at fault for the confusion, though each party concerned could have handled things better. Hoffman should have stated clearly that the Validate program would delete itself after the year 2000, thus avoiding suspicion that it had been trojanised.

It would be interesting to know how many users (or indeed anti-virus researchers) routinely used *VSUM* on their machines without ever realising that the programs might do things which they never expected? This incident serves to underline the dangers of blind trust in software.

However, most of the problems lie at *Central Point's* door. While it is acceptable (or possibly even laudable) for *CPAV* to detect these self-deleting files, it is vital that the user is told precisely *what* the package has found. The term 'computer virus' is now so emotive that it should be used correctly or not at all.

Technical Support Issues

Several days after the incident both the Technical Editor and the Editor of *VB* called *Central Point's* technical support line independently, and both were told that the identification was real and that the Validate program was infected by the 'Perry virus'. This was after assurances to Hoffman that the problem had been dealt with.

While speaking to 'Doug' (who explained that Perry was 'a virus, but it doesn't spread real much (sic)') the Editor was advised to 'upgrade to CPAV 2.0' so that the program could be disinfected. *Central Point* carries *VSUM* on its BBS - so how was the problem not spotted?

"An increasingly important part of the anti-virus service is technical support - is Central Point's good enough?"

Issues

Fortunately, *Central Point* was alerted to the problem before duplicating the signature strings for the *MSAV* product - had this not been the case, the problems could have been even larger. With *MSAV* so widespread, the damage caused to an innocent product's reputation could have been disastrous. The opportunities for litigation must be rife, and doubtless lawyers worldwide are rubbing their hands in glee at the thought of the first widespread *MSAV* false alarm.

The problem for the anti-virus vendors when faced with programs like Perry is that if they choose not to detect them, it is likely that they will fall behind in reviews which test against arbitrary collections of malicious code - ironically, *Central Point* may well have included the Perry detection routine in a specific effort to improve its *VSUM* rating.

False positives are set to become the big issue in the anti-virus industry. It is paramount that if a false positive occurs, users are informed in no uncertain terms. While *Central Point* would debate whether this result is truly a false positive, there is little doubt that their reaction was too slow.

The position of the industry seems to be that there is nothing wrong with detecting the presence of the Perry-like code within files, but that this code should be explained for what it is, and the user has to be made aware of the actual threat posed. If potentially malicious code is detected, the anti-virus software manufacturer should explain to the user what that code does - the label 'infected' is not enough. An increasingly important part of the anti-virus service is technical support - is *Central Point's* good enough? ∎

IBM PC VIRUSES (UPDATE)

Updates and amendments to the *Virus Bulletin Table of Known IBM PC Viruses* as of 21st August 1993. Each entry consists of the virus' name, its aliases (if any) and the virus type. This is followed by a short description (if available) and a 24-byte hexadecimal search pattern to detect the presence of the virus with a disk utility or preferably a dedicated scanner which contains a user-updatable pattern library.

Type Codes

C	Infects COM files	**M**	Infects Master Boot Sector (Track 0, Head 0, Sector 1)
E	Infects EXE files	**R**	Memory-resident after infection
D	Infects DOS Boot Sector (logical sector 0 on disk)	**P**	Companion virus
N	Not memory-resident	**L**	Link virus

_185 (temporary name) **CR:** A 185 byte virus which does nothing but replicate.

```
_185            5350 593D 004B 755C 561E 5053 5152 B802 3DCD B372 4993 B43F
```

_198 **CN:** This small virus reboots the machine if an infected program is run on a Sunday. It seems to be related to a nameless 205 byte virus.

```
_198            215E 803C B874 3D33 C933 D2B8 0042 CD21 832E 0901 0DBA 0001
```

Anti-Pascal_II.401 **CN:** This variant is closely related to the 400 byte version, only one byte longer. It is detected with the Anti-Pascal_II pattern.

ARCV.Christmas **CN**: A few of the viruses created by the (now defunct) ARCV group have not been listed here before. Christmas is 670 bytes, and as the name implies it atcivates around Christmas, displaying the text 'Happy Xmas from The ARCV Made in ENGLAND [ARCVXMAS] by ICE-9 Released June 1992'. Slime (CR, 773 bytes) and 916 (CR, 916 bytes) use slightly polymorphic encryption. The 1060 variant (CER) has also been named ARCV.Twins.

```
ARCV.Christmas  2E30 1547 E2FA C390 5053 5152 2E80 847E 0302 E86E FFE8 DEFF
ARCV.Slime      BE?? ??BD 89FE E2FE 2E81 ???? ??46 4645 75F6
ARCV.916        BE?? ??B9 BF01 2E81 ???? ??83 C602 4975 F5
ARCV.1060       E800 005F 81EF 0701 E802 00EB 128D B523 01B9 B603 8CC8 8ED8
```

Cascade.1701.G **CR:** This virus is a slightly modified version of one of the earlier variants, and most virus scanners will probably not notice the difference. Detected with the Cascade (1) pattern.

Cascade.1701.Jojo.C **CR:** A 1701 byte variant, which is detected with the Jojo pattern.

CPW.1527 **CR:** This virus from Chile was originally called 'Mierda', but that name has now been rejected. The virus may be related to an earlier CPW virus, or it may have been written by the same author, but the exact relationship has not been determined. The virus is encrypted, 1527 bytes long, and contains the text message: 'CPW fue hecho en Chile en 1992, VIVA CHILE MIERDA!'. The virus targets several anti-virus programs, including *Central Point Anti-Virus, Dr Solomon's Anti-Virus Toolkit* and *McAfee Associates SCAN*.

```
Cpw.1527        E800 005F 83EF 038B F7F9 E805 02E9 FE00 0E1F 33F6 33FF F8E8
```

CyberTech.503 **CN:** Similar to an earlier 664 byte variant, but only 503 bytes long.

```
CyberTech.503 E800 005D 83ED 0750 8DB6 1B00 89F7 B9E0 01AC 34?? AAE2 FA
```

Dark Avenger.1800.Sneaker **CER:** A minor variant, where the message at the beginning has been changed to 'Nadia FOTTITI! By The Sneaker'. Detected with the Dark Avenger pattern.

Dosver **CER:** An encrypted virus 2062 bytes long. Awaiting full analysis.

```
Dosver          5156 1E0E 1FB9 6406 FA30 2446 E2FB FB1F 5E59 C3
```

Dy **CN:** A simple virus, 278 bytes long.

```
Dy              803E 0000 5A74 03E9 9600 812E 0300 1200 7303 E984 00A1 1200
```

Eddie-2.D **CER:** This version seems to have been created by changing a single 'mov cx, ss' instruction to the equivalent 'push ss, pop cx'. As this instruction was in the middle of the search string used by VB as well as McAfee's SCAN, this is almost certainly an attempt to avoid a specific scanner.

```
Eddie-2.D       D3E8 4016 5903 C18C D949 8EC1 BF02 00BA 2B00 8B0D 2BCA 3BC8
```

Faerie

CN: This is a simple, 276 byte virus which does nothing but replicate. It contains the word 'Faerie'.

```
Faerie          CD21 33D2 33C9 B802 42CD 212D 0300 8986 0401 8986 0F02 8D96
```

Golgi

CER: This is one of the few viruses that were first discovered 'in the wild' - in this case the first incident was reported in Canada. The virus is 605 bytes long and contains the text '[Golgi Testicles] v2.0.Copyright (c) 1993 Memory Lapse'. The virus has not been fully analysed, but appears to do little but replicate.

```
Golgi           3D3D 7414 3D00 4B74 5580 FC11 740C 80FC 1274 07EA
```

HideNowt

CER: A 1741 byte virus, which uses several anti-debugging techniques [*For a full analysis of this virus, which is in the wild in the UK, see p.9. Ed.*].

```
HideNowt        5F0E 0726 8C55 1026 8965 1257 B910 0033 F68E DEF3 A458 FABC
```

Icecream

CN: This 501 byte virus is probably written by the Dutch 'Trident' group, and may be related to the Cybertech viruses. The virus uses self-modifying encryption, and cannot be detected with a simple search pattern.

Jerusalem

CER: Three new 'no-name' 1808/1813 byte variants have been reported this month. Two of them are detected with the Jerusalem-US pattern, and the third is detected with the Captain Trips pattern.

Leprosy.1547, Leprosy.1874

CEN: Two variants of this family of overwriting viruses, 1547 and 1874 bytes long. Both are detected with the Silver Dollar pattern.

Metallica

CR: A 1739 byte virus. Awaiting analysis.

```
Metallica       86E0 3C3D 7417 3C4B 7413 3C43 740F 3C56 740B 3CFA 7413 86C4
```

Murphy.HIV.B

CER: Just like the original HIV variant, this virus is 1614 bytes long, but has been slightly modified. Detected with the Murphy_2 pattern.

Pixel.296

CN: Closely related to the Pixel.277 virus, and detected by the same pattern.

Pixel.1271

CN: Another new variant which is detected by an old pattern - in this case the pattern published for the Pixel.936 virus.

Russian Tiny.145

CN: This virus is related to a 146 byte variant described earlier, and is detected with the same pattern.

SatanBug

CER: A variable-size virus, around 5000 bytes long, that uses self-modifying encryption. The virus can not be detected with a simple search pattern.

Shanghai

CR: An 848 byte virus. Awaiting analysis.

```
Shanghai        9C80 FCBB 7505 B811 119D CF80 FC3B 740E 80FC 3674 099D 2EFF
```

Syslock.Cookie.B

CER: This variant has a slightly modified decryption routine, probably in order to avoid a specific scanner, possibly one relying on the pattern published in the *Virus Bulletin*. The virus is 2232 bytes like the original, and it also displays the text 'I want a COOKIE!'.

```
Cookie.B        8AE1 FB8A C133 0614 0031 0446 46E2 F15E 5958 C3E8 DFFF CD21
```

Tomato

CER: This 2156 byte virus has not been fully analysed, but contains some of the silliest text messages ever seen in a computer virus, including 'have you ever danced with the devil under the red light of a big tomato?' [*A misquote from the film 'Batman'. Ed.*]

```
Tomato          B80D 0150 FBBF 0D01 2E8B 3630 002E 8B05 33C6 2E89 0547 4781
```

Trivial.177

CN: This overwriting, 177 byte virus has also been called Good Thursday, but it is not considered significant enough to deserve a separate family classification.

```
Trivial.177     B40F CD10 32E4 CD10 BE83 01B4 0EAC 0AC0 7405 CD10 EBF7 C3B8
```

Vienna.827

CN: This 827 byte variant is detected with the Violator pattern. Two other Vienna variants, both 648 bytes long, have been found recently, both closely related to the Lisbon variants. They are detected with the Vienna-3 and Dr. Q patterns.

Viruz.729

CN: A primitive South African virus, that overwrites the first 729 bytes of COM files, and then appends 39 bytes to the file. This virus contains code to format the hard disk.

```
Viruz.729       B900 01B8 0807 CD13 B000 E670 E671 FEC0 EBF8 B419 CD21 3C00
```

Wanderer

CR: This 400 byte virus contains the text 'As wolfs among sheep we have wandered'.

```
Wanderer        80FC 4B75 03E9 6300 80FC 4E74 2F80 FC4F 742A E9CF 0020 4173
```

Willow

ER: This 1870 byte virus has not been fully analysed, but one interesting feature has been observed - different samples of the virus have different entry points, perhaps in order to confuse certain anti-virus programs. The main effect of the virus is to delete COM files when they are executed.

```
Willow          B442 CDFD 7204 5B59 5DC3 BAFF FFB8 FFFF EBF4 558B EC1E 5657
```

INSIGHT

The Husky Voice of Reason

Mark Hamilton

Sophos is one of the major producers of anti-virus software, yet that is only one facet of its operations: a large proportion of *Sophos'* work is undertaken subject to highly confidential agreements and can not be discussed or written about. Fortunately, its technical director was persuaded to leave the seclusion of his Oxfordshire hideaway temporarily and talk to me about his own very personal view of the anti-virus industry.

Variously known as 'Mad Dog' (a name coined by Edward Wilding, *Virus Bulletin's* erstwhile editor) and 'Husky' (his *CIX nom de plume*), Dr Jan Hruska was educated at Cambridge and took his doctorate at Oxford. After such an auspicious start, how did he get involved in the murky world of computer viruses? 'Pure ill-fortune' jokes Hruska. 'At Oxford, I had to make a decision whether or not to go into the corporate world and was unlucky enough to meet Peter Lammer. We decided to go into business together, and thus *Sophos* was started.'

A Secure Start

For the first five years of its life, *Sophos* designed hardware devices, principally for telephone call logging. 'We chose to go into software in around 1985 and, in order to finance development, we raised some venture capital which involved becoming incorporated in 1987' he explained.

Why did he decide to develop security software? 'We saw that the use of PCs was going to lead to security problems as they became more widespread. Nobody knew about viruses in those days; we concentrated on encryption and data confidentiality above everything else. An early breakthrough came when *Sophos* won the contract to supply security software for the Government Data Network, GDN.'

'When viruses came along, we adapted a program which was originally written for auditing purposes. The program was modified to check the integrity of the Operating System. Viruses corrupt integrity and so we already had a product which could detect this infection.' The product (renamed *Vaccine*) has since been completely rewritten and gone through a number of versions.

A Clean Sweep

Sweep is one of the few anti-virus products that does not have disinfection capabilities. I asked Hruska when he decided on this policy and his reasons for its continuance. 'After seeing parasitic viruses in various mutated forms, it was clear that modifications to viruses would be made at fairly regular intervals. It was quite obvious that chopping one byte too many off a program - or leaving one byte too many inside a program - could be the difference between that program crashing and not crashing and possibly causing far worse corruption than the virus would have in the first place' explains Hruska.

There is no question in Hruska's mind as to the veracity of this stance. 'We decided that the only right policy was to replace the programs with the originals. I am quite convinced that doctors would have an easier time if they could kill the patient and restart with an uninfected copy but unfortunately you can't do this! But you *can* kill computer programs and start anew. It's cheap and it's quick, and above all else, it is secure.'

Hruska on Solomon: 'We have been trying to push each other off the cliff for quite a long time, so far without much success, and I intend to continue doing it!'

Biology Lesson

Hruska believes that the demand for the disinfection of infected files is market driven. 'I think that the medical parallels between computer and biological viruses have led to the market demand for cleaning. No amount of persuasion that the job can be done much more thoroughly and much more reliably in a different way makes the market behave otherwise. The consequences of operating corrupted programs, which in turn produce corrupt data for a long period of time, are a nightmare. All you need inside a program is one *bit* that is wrong.'

Does he believe that users will eventually wake up to these facts? 'It's a twofold problem' says Hruska. 'The first one is that the disinfection part of the program is only invoked once the virus infection happens and "once the virus infection happens" is not often for the majority of users. So the users themselves don't end up testing it.'

'Disinfection gives users some hope as far as their executables go and if that's not successful, they will just accept the failure with a shrug of the shoulders. We have never been in that game!' adds Hruska emphatically.

Hruska believes that many smaller users buy their anti-virus software in much the same way that they buy other consumer goods such as washing machines: they compare each product's features list and go with the one with the most ticks. 'If one of the ticks is 'disinfection', they will demand it. Unfortunately, the second problem with disinfection is that there is no quality assurance performed by the virus writers, so there is no guarantee that a virus will perform infection correctly. If the infection is not done correctly, I can't see how disinfection could be performed in all cases.'

"Disinfection gives users some hope as far as their executables go and if that's not successful, they will just accept failure with a shrug of the shoulders."

Sizeable Problems

Hruska is concerned that as the number of viruses rise, scanners will become increasingly unmanageable under DOS. 'Where we are going to get into problems with scanners, at least DOS-based scanners, is the fact that, whether you like it or not, on average you need about thirty bytes of data to describe a particular virus. You can do clever things like compressing the data, but mainly it's fairly rich data, hence it can't be compressed a lot. If you multiply that by 10,000 - the number of viruses we will have in the forseeable future - the amount of memory required starts to become non-trivial.'

The way ahead is not to run scanners on the workstation, but on the file server, says Hruska. 'One can see that it is the sheer volume of information that is going to be a problem, which is why our strategy has always been to go towards other platforms and put scanning on the network server. Hence our concerted effort to produce scanners for *Netware*, *OpenVMS* and *OS/2*. Those will be the file servers of the future and that is where we can continue the expansion more or less indefinitely.'

A Tight Fit?

Why does *Sophos* not simply take advantage of the extended and expanded memory most machines have? 'The great burden that DOS software manufacturers carry is the lowest common denominator, which means that all the software has to be written to run on the humblest *Amstrad PC* equipped with an 8086 chip and 640k RAM.' I pointed out that *Central Point* has effectively ceased supporting 8088 and 8086-based PCs with the latest release of its software. '*Central Point* is not a competitor', he retorted, with an amused glint in his eye, referring to Jim Hornsburgh's comments on Sophos' turnover (*VB* August 93, pp. 7-8).

Windows Windows Everywhere...

'I sometimes suspect that there is actually a great conspiracy between large software manufacturers and hardware manufacturers each fuelling each other's research so *Microsoft* can conveniently forget about the 8088 and *Intel* can push out yet another processor', Hruska explains. 'That in turns gives the software manufacturers the ability to develop more complicated software and so on.'

Hruska believes that the push to Pentium power is largely unnecessary. 'In most cases a 286 processor is more than sufficient for everyday use. It was only when *Microsoft* convinced users that all we wanted in life were pretty pictures on the screen that it became too pedestrian. Where it's all going to lead, I'm not entirely sure. There are a hell of a lot of 8086-based processors around and, as long as somebody is using them, 360k 5.25-inch and 720k 3.5-inch disks must be the standard means of delivery of software. The software must run within 640k of RAM.'

'We have done an informal survey asking a number of users if 1.44M disks would be all right. The answer was a resounding "No". We have also asked users whether 360k disks were required and the answer was a definite "Yes". Hence all our software is delivered on 360k and 720k disks. I can't see that that is going to change in the near future.'

Lemmings

I reminded Hruska of Dr Alan Solomon's boast that he'll be the last to 'jump off Beachy Head' - meaning that his product will keep to virus-specific detection long after all his competitors have packed up. Will *Sophos* keep to this methodology too and for as long? 'Do you mean jumping off the cliff together? Embraced?!' he exclaims. 'We have been trying to push each other off the cliff for quite a long time, so far without much success and I intend to keep on doing it. I am sure that this is a feeling which he shares - it is purely a matter of competition. As for the sentiment that he'll be the last person: well, he's welcome to his opinion.'

Hruska agrees with Solomon's belief that companies that have bought in anti-virus technology to badge and sell on, simply won't be able to keep up in this fast moving marketplace, and points to the demise of *XTree's* product as evidence. 'I think that in order to be successful in the virus-specific market, you really have to do research in-house. From that point of view, I think that the specialists will always come out on top.'

When he's not writing software, disassembling viruses or lecturing, he likes to relax by playing the piano. He also enjoys scuba diving and flying light aircraft. A man of many talents, it seems - and Hruska intends to use all of them to stay ahead of the game.

VIRUS ANALYSIS 2

Nothing To Hide?

Jim Bates

This month's report concerns a virus which was said to be at large in the US last year but has recently been reported at a Welsh University in the United Kingdom. The virus has been named HideNowt (for some unknown reason) but is also known as 1757. It is a typical parasitic virus which infects COM (including COMMAND.COM) and EXE files by appending the virus code and modifying the file header.

A Stack of Problems

I have previously noted that virus writers seem to have a penchant for taking a single idea and overworking it *ad nauseam* within a particular creation. HideNowt is a classic example of this and the obsession in this case involves the use of the stack. The writer tries to complicate his code by manipulating the contents of the stack in a way that is presumably designed to confuse anyone trying to disassemble the virus. There is no other valid technical reason for manipulating the stack in this way, and it provides ample opportunity for the virus author to make even more coding mistakes than usual.

So let us examine the virus in detail. HideNowt is a memory-resident parasitic infector which targets COM and EXE files longer than 1025 bytes. The virus is 1741 bytes in length but because of a rounding up routine, the length added to infected files will vary between 1741 and 1756 bytes depending upon the original length of the host file. The code is partially encrypted by a woefully inadequate fixed encryption routine which presents no problems to software developers, as detection by a simple pattern recognition routine is possible. There is no attempt to hide its presence in infected files, but HideNowt does use primitive tunnelling to try to gain clean access to the system services. There is also a depressingly inept attempt at armouring via stack manipulation.

Installation

The entry point for this virus is the same regardless of whether the infected host file is of COM or EXE type and processing begins by turning off the interrupt enable and trap flags. The Interrupt Mask Register is then accessed directly (through its Port) in order to disable the servicing of all Non-Maskable Interrupt request lines.

Having effectively isolated the processor from its peripheral hardware, a routine is called which takes a copy of the first four interrupt vector addresses in low memory and then sets up the stack pointer registers to use the low memory area. Theoretically this will disable any attempt to debug the subsequent code since the low interrupts (Int 01h and Int 03h) are used for debugging purposes. Virus researchers are wise to such techniques and can easily circumvent them, but this simple trick may confuse those who simply want to 'play' at disassembling or reverse-engineering the code.

Processing then continues with a routine which calculates a segment value that allows the virus code to function from a relative offset of 100h. Once again, this type of manipulation is unnecessary but it does enable the author to think in standard terms. Once the offsets have been reset successfully, a small routine decrypts an earlier portion (288 bytes) of code before the stack position is reset, the low memory part of the interrupt table is repaired and normal interrupt processing is re-enabled. The manner in which this is accomplished may cause system malfunction on certain machine configurations.

The code then uses the Int 01h/ Int 03h method of attempting to strip back the DOS Int 21h service routine [*This process is known as tunnelling. Ed.*] until a clean entry point is found. The routine which does this is not very effective and the virus may be forced into using the ordinary vector address. Processing goes on to decrypt data near the end of the virus code which contains the filename C:\COMMAND.COM, which the virus attempts to infect.

Only at this point does the code attempt to determine whether the virus is already memory-resident. This is done by obtaining the segment address of the current Int 21h vector and checking that a value of 7859h exists at offset 07C7h within that segment. Thus if some other program adds a legal vector to the Int 21h routine (quite likely once COMMAND.COM is infected), the virus will become installed (and active) twice in memory. The provision of certain checks within the code seem to indicate that the writer was aware of this possibility and attempted (unsuccessfully) to exploit it.

If the recognition word is not found the virus is installed in high memory and hooks interception routines into the Int 21h vector and the DOS CP/M entry point. The relevant memory pointers are changed to allow the code to remain resident and undisturbed. Processing then finally replaces the original entry pointers before passing control into the host program.

Infection

When resident, the virus intercepts requests for the old CP/M services as well as the more usual Int 21h functions. The CP/M services are a throwback to the time when MS-DOS needed to provide compatibility with certain CP/M function requests. There are still some programs which require these functions and so they are maintained.

The virus intercepts the request by checking the subfunction request (stored in the CL register) and discarding any below a value of 25h (returning directly to the calling routine). Other functions are redirected by restructuring the stack into a DOS acceptable format and placing the CP/M subfunction into the AH register before passing control to the normal DOS handling routine. The virus Int 21h handler only intercepts functions 11h, 12h and 1Ah.

Function 1Ah is a request to change the address of the Disk Transfer Area. The virus interception simply collects the new address and stores it within its own code before allowing the request to continue normally.

Functions 11h and 12h are requests to Find First and Find Next file using the old style File Control Block methods. These functions are severely limited on modern machines but are still used extensively by some internal DOS services. The virus code only attempts to infect a file found during the Find First interception. Once a file has been infected a flag is set within the virus to prevent any further infection until the next Find First call.

Corruption and Damage

As mentioned above, actual infection is unremarkable and consists of appending code to a suitable target file and modifying either the initial three bytes (with COM files) or modifying the header contents (EXE files). The usual precautions of re-vectoring the DOS critical error handling (Int 24h) and the Control Break status are taken. Target file's date/time settings and attributes are preserved during infection but for some reason the virus writer has included code to check (and turn off) the archive bit of an infected file if its attributes were not set to Read Only.

It should be noted that the virus relies upon the file extension to determine its type and this will result in corruption of those files with incorrect extensions (eg *Windows* specific program files, converted files etc.). Another error in the program will cause irreparable corruption of COM files longer than 63794 bytes. No attempt is made to hide the growth in file length and this will be noticed as an increase of between 1741 and 1756 bytes.

Letters Game

One final point worthy of mention is that during the file infection routine, a check is made of the target filename so that infection of certain files can be avoided. The manner of the check is such that hundreds of thousands of possible filenames would match the criteria and so I present the checking routine here in its original form together with the check values.

Concerned users and vendors can thus check to see whether their own files are involved, as the test is very easy to make. The preceding code points the SI register to the filename in the simple form of filename.ext (the dot exists in the buffer as ASCII character 2eh) and the buffer is padded with spaces up to a length of 11 bytes. CX contains a loop count of 4 and BX contains 0. The summing routine consists of the following sequence of instructions:

```
Summer:  LODSW
         ADD  BX,AX
         LOOP  Summer
```

The total returned in BX is then checked for one of five separate values: 201Dh, 1C1Ah, 4A46h, 1C11h and 1810h. If the summed target filename matches any of these values, it is *not* infected.

Conclusions

In the projected scheme of things from some people in the anti-virus industry, this virus would be classified as 'non-malicious' code because it contains no trigger routine and no payload. However, it is every bit as insidiously destructive as the most vicious virus programs quite simply because its frequent blunders will damage user confidence along with the data and programs that it corrupts.

With the continuous publicity across the industry about the damage caused by virus code, it is no longer possible for virus writers to claim they were 'only doing it for research' or 'didn't intend any harm'. I am compiling detailed files of known virus writers so if anyone can positively identify a known virus writer, please let me know.

HIDENOWT

Aliases:	1757.
Type:	Resident, Parasitic infector.
Infection:	COM and EXE files (including COMMAND.COM).
Self-Recognition in Files:	
	Infected files contain 7859h as the first of the last three words in the file.
Self-Recognition in Memory:	
	Value of 7859h at offset 07C7h of the DOS INT 21h entry segment.
Hex Pattern:	
	33F6 8EDE F3A4 58FA BCFD FF8C DB8E D303 E69C 0E05 1A00 0E1F
Intercepts:	INT 21h Fn 11h and INT 1Ah for internal use by the virus. INT 21h Fn 12h for infection.
Trigger:	No trigger routine but will cause file and system corruption.
Removal:	Specific and general disinfection is possible but it is recommended that infected files be identified and replaced under clean conditions.

VIRUS ANALYSIS 2

NukeHard - Starting With a Bang?

Eugene Kaspersky

As the war between the anti-virus software manufacturers and the virus writers continues, both sides attempt to adopt new strategies which will give them the upper hand. While the virus hunters search for new algorithms to detect viruses more accurately and quickly, the computer underground is attempting to develop its two principal weapons against anti-virus software: stealth and polymorphism.

One of the biggest problems which faces the scannermanufacturers is the performance impact caused by searching executable files for polymorphic viruses - and this is a problem which is getting worse. The first polymorphic viruses found were discovered about three years ago (V2P2 and Phoenix) and since these early attempts to foil the scanner manufacturers, there have been several dozen more polymorphic viruses written.

Nothing Like a DAME...

A recent addition to the armoury of the would-be virus writer is the distribution of so-called 'polymorphic' engines as object modules. This linkable code allows inexperienced programmers to produce complex self-mutating code. The first example of such a beast was the Mutation Engine, which was distributed as an object module, along with an example virus (Dedicated) which uses MtE encryption.

The MtE was designed so that it could be linked in to a virus (or any program) simply by being called with several simple parameters. The resulting virus would have two separate parts: the encrypted code of the virus, and the encryption/decryption routine which is used to decrypt the virus code proper. When such a virus is executed, the decryption routine first decrypts the virus code and passes control to it. When the virus infects another file, the encryption routine is called, which 'wraps' the virus in a decryption routine produced by the MtE, effectively hiding the virus infection.

The MtE is designed to produce a different decryption routine every time it is run. This means every infection of a virus which uses the MtE 'looks' different, making viruses which use it very difficult to detect. Fortunately there are several limits to the MtE which enable the construction of an algorithm capable of detecting MtE encrypted files. The first shortcoming of the MtE is that the body of the decryption routine always consists of one loop. Secondly, the generator uses word registers exclusively (with the exception of several instructions which use the CL register). These fixed features allow a set of detection criteria to be established.

One unusual feature of MtE encrypted files is that occasionally the code produces an unencrypted file, meaning that the body of the Mutation Engine is unencrypted in the host file. A second 'feature' of the MtE is that it contains several bugs and sometimes produces flawed decryption routines which causes the computer to hang when infected files are run.

The next important event in the history of polymorphic viruses was the distribution of a more sophisticated polymorphic engine, the Trident Polymorphic Engine (TPE). There are four different versions of TPE, and each is more complex than the Mutation Engine. Girafe and Civil War are both examples of TPE-based viruses.

Viruses encrypted by TPE are more difficult to detect as TPE is capable of creating different loops and uses more instructions in the decryption routine. Again, like the MtE, it has its limits - in this case that the encryption algorithm is always based on a simple XOR or ADD technique. Like most code produced by members of the Computer underground, TPE is littered with bugs. Some versions of TPE generate decryption routines which cause the host machine to crash by generating a hardware interrupt.

The latest encryption device in this sordid tale has been written by 'Nowhere Man' (the author of the less-than-reliable Virus Creation Laboratory). Named Nowhere Man's Encryption Device, or NED for short [*Here we go again... Ed.*], the polymorphic engine is an attempt to thwart the scanner developers, though from analysing the engine it has to be deemed a failure.

Overview

NukeHard is a 1795 byte parasitic file infecting virus which uses NED. The virus is not memory-resident, and therefore uses the 'single shot' infection mechanism employed by most other such viruses.

When an infected file is executed, control is passed to the virus code by means of a JMP instruction which was inserted at the start of the host file. A routine decrypts the body of the virus and passes control to it. Next, the virus code repairs the image of the host file in memory, by replacing the overwritten bytes at the start of the program.

Processing then searches for two files which have the extension COM. For these files to be suitable for infection, the only criteria which they must match are that they are shorter than FD49h bytes (64,841 bytes) and that they are not already infected. In order to ascertain this, the virus code examines the first byte of the target file and, if this is E9h, assumes that the file is infected. If the target file is deemed suitable, it is infected and the virus code encrypted. The length of the encryption routine is not constant, so the length by which the host file grows is not fixed.

PS-MPC

The PS-MPC (Phalcon/Skism Mass Produced Code Generator is one of a number of virus construction toolkits which are available on many virus exchange BBSs. PS-MPS was first examined in *Virus Bulletin* in Autumn of last year (*VB*, September 1992, p.3) and is capable of producing *TASM*-compatible *Intel* 8086 assembly language which can be compiled to produce fully-functional viruses.

Early versions of PS-MPC were confined to infecting only COM files, and viruses produced were not capable of becoming memory-resident. However, later versions of PS-MPC allow the creation of memory-resident COM and EXE infectors.

The part of the virus code which hanldes the replication of the virus to replicate is very simple, and contains a number of errors. It does not intercept the DOS Critical Error handler during infection, and attempts to infect write-protected disks will result in the familiar 'Write protect' error message. In addition, the internal structure of the file is not checked, so errors occur if the file is not in the standard COM format.

A December Format

Before the virus returns control to the host file, it checks the current date, using the DOS function Int 21h with AH=2Ah. If the system date is set to December (any day) the trigger routine is called.

The trigger routine is intended to reformat the contents of the hard drive, using Int 13h, function 05h (Hard Disk Format), and then reboot the computer. Fortunately, like so many of the destructive trigger routines contained in viruses, the routine does not function as its author intended.

Reformatting a hard disk is not the trivial task virus authors often assume. It is not sufficient simply to call the correct Int 13h function- a number of registers and pointers need to be set or else the format will fail. Fortunately the virus author has clearly failed to test his creation, as he has made just this error when writing NukeHard, although under certain circumstances the format will be successful.

Nuke Encryption Device

The encryption engine which this virus uses has clearly been written by a different author to the body of the virus, which looks as though it was produced using the PS-MPC virus construction toolkit. The entire virus has the feel of two different pieces of code which have been linked to form one.

The replication code is typical of that found within viruses constructed using PS-MPC - the virus does not hooks Int 24h and there is an error on Int 13h call in trigger routine. The encryption engine, however, has clearly had a great deal of time and development put into it.

The way NED is called is reminiscent of both the MtE and TPE. NED requires certain parameters in order to instruct it which areas to encrypt. These are passed in the registers AX, BX, CX, DX and SI as shown below:

- AX points to the buffer into which the encrypted code should be copied.
- BX contains a pointer to the first byte of the code to be encrypted.
- CX contains the offset of the first byte of the virus.
- DX contains the length of the infection code.
- SI is used to set various options within NED.

There are two text strings in the virus body. The first is

```
IT'S HARD. SMAUG REIGNS SUPREME AS THENEW
FORCE IN VIRUS WRITING. SMIRK. SMIRK. POOF!
```

This is found in the replication section of the virus. The second text string is

```
[NuKE] Encryption Device v1.00 (C) 1992
Nowhere Man and [NuKE]
```

and is placed in the polymorphic generator code, adding further evidence that the two parts of the code were written by different people.

Conclusions

This virus is the end-product of combining a new encryption engine with a machine-produced virus. It is difficult not to wonder how long it will be until there is a whole family of viruses which use this engine.

NukeHard

Aliases:	None known.
Type:	Not memory resident, Parasitic file infector, Polymorphic.
Infection:	COM files only.
Self-Recognition in File:	Checks the first byte of the file body for the value E9h (JMP to virus body instruction).
Self-Recognition in memory:	None.
Hex Pattern:	No search pattern is possible.
Intercepts:	None.
Trigger:	On December it attempts to format the fixed disk drive.
Removal:	Under clean system conditions identify and replace infected files.

VIRUS ANALYSIS 3

Nervous Twitch

James Beckett

Life gets ever more interesting... or so some people keep trying to tell me. Most viruses are certainly pretty uninspiring and though I have to disassemble the things from time to time, I try to avoid writing about them (as the Editor will ruefully confirm). The occasional interesting one provides more to think about even if it does represent another few late nights spent at the office.

Researchers have been expecting viruses aimed at systems other than DOS for some time, the limiting factors being simply the availability of knowledge about the systems to be attacked, and having them used widely enough to be worthwhile targets. The relatively simple DOS internals have been laid bare by hundreds of books and magazine articles, but the more complex systems of *Windows*, *OS/2* and now *NT* are much less well understood.

MS-Windows has been around for a long time now in some incarnation or other, and we are only now seeing viruses aimed at it; hopefully the opacity and protection afforded by bigger systems will improve their life-expectancy.

Open Windows

Twitch is the second known *Windows* virus, but it is the first to be fully *Windows*-aware rather than just, as it were, *Windows*-compliant. WinVir1.4 (*VB* November 1992, pp.19-20) explited the NE-format EXE file header used for *Windows* executables, and infection copied part of the host code to the end of its file just as a normal COM infector does. However, fixing up the host and running it immediately was beyond WinVir's limited ability, the author choosing instead to disinfect the file to enable its execution on a second attempt.

Twitch is more transparent, and is properly *Windows*-resident, unlike the one-shot infection mechanism of WinVir. It was written in a high-level language, the usual way of writing *Windows* programs, probably C considering the library routines attached to it. (One problem when pulling it apart was distinguishing between virus code, C runtime code, and *Windows* library code.) On activation the virus disappears into the background of the system and goes about its infection with the user none the wiser.

Under DOS, standard applications run in transient mode - they are loaded into memory and run until terminated, assuming control of all available resources. Viruses generally either do this for a very short period, infecting say a single file each time before passing control to the host program, or become memory-resident - staying loaded but gaining control of the system on a periodic, non-intrusive basis. The standard way of doing this is through the TSR interface, which allows the 'background' operation of a program while other transients are running. Viruses tend to do this in slightly dubious and error-prone ways, but the result is largely the same. Thus many virus descriptions refer to 'Resident' or 'Non-resident' status.

The situation under *Windows* is very different - every program is in effect resident, except they are managed and allocated resources by the supervising program, *Windows* itself. 'Resident' and 'Non-resident' merge at this point, and it is just a question of how long the program takes to do its task. Programs may have parts moved between memory and disk when not in immediate use, with only *Windows* knowing that this underhand behaviour is going on. More memory is available for applications, and because several things may be happening concurrently, additional operations can easily go unnoticed - such as the slow infection of executables by a virus.

Windows still relies on DOS for much of its operation, thus inheriting many of its defects and inadequacies, including its lack of any form of user file protection scheme. There is no reason why a virus running as a Windows process cannot infect every file on a disk, or indeed cause damage in a trigger function.

Analysis

The virus will be initiated by any method of running the host program - by clicking on a Program Manager icon, clicking the name in File Manager, or using the Run command in the File menu of either.

On starting the infected program, an 'invisible' window is created. This accepts no keyboard input or mouse input, but it does set up a timer callback - a request for *Windows* to repeatedly inform it when a certain time interval has elapsed. After this, it simply runs the original program, retained on disk as an OVL file from infection time. The user is unlikely to notice the little extra time taken.

Windows duly calls the virus every sixty seconds and the next target is picked for infection.

Twitch implements something known as a State Machine - often used in compilers and text processing languages for matching patterns in files, here it just keeps track of what to do next. At each call, an action is performed depending on the current state, and the result of the action defines the next state. The virus starts off by modifying an entry in the WIN.INI file as a flag for future reference. If you see a line 'DeviceSelectedTimeout' in your WIN.INI (cunningly disguised to look like the legitimate 'DeviceNotSelectedTimeout') then the virus is looking over your shoulder. As well as later infecting existing files, the virus adds up to four completely new files to try and ensure its survival - an infected NETWARE.EXE, FILEMAN.EXE, SCRNSAVE.EXE or WINPRINT.EXE is created in the Windows directory, and its name inserted into the 'load=' section of WIN.INI in order to ensure that the virus is loaded whenever *Windows* is started.

The next two states infect every *Windows* executable in the *Windows* directory at a rate of one per minute, with the exception of the shell (usually Program Manager). Once all these files are infected, the virus infects the files SYSEDIT.EXE and NWPOPUP.EXE in the *Windows* system directory. The PATH is the next target - again, any *Windows* executables found are infected.

After all this, it stays in the last state, where it eats some system resources and interferes with the screen. The name Twitch comes from the side-effect at this point, which is to cause every window on the screen to redisplay its data. Normally, when a window is uncovered by another application, the newly uncovered window is sent a message by *Windows* so that it can restore itself. However, other applications can send these messages to windows as well, and Twitch sends the redisplay message to the entire desktop. The rippling result is unmistakable, and distinctly disturbing to the eyes.

While running, no *Windows* functions are intercepted to infect programs as they are run, in the manner of a DOS virus, so unless drive A: or B: is in the PATH (not a usual configuration) programs on floppy drives will not be infected. This will probably limit its spread. Furthermore, as with other companion viruses, copying an infected file to another disk is likely to give the game away, as only the virus part will be copied, with the overlay file containing the original program being left behind.

On the other hand, these days packages are often composed of several files - frequently a package will have additional files with the extension .HLP, .INI and .DLL. Typically one would pass a package to someone using COPY PLAYER.* A: or just COPY *.* A:. An extra OVL file would be overlooked or considered natural. As it happens, Windows programs don't need separate overlays as everything can be loaded from the executable and discarded as required. This also means the virus is not going to reveal its presence accidentally by creating its OVL file over an existing one and destroying part of the target program.

Central Point Targetted... Again

An unexpected ability of the virus is that it includes code to subvert *Central Point's* checksummer: on every infection, to avoid detection by *Central Point's* checksummer the virus deletes the file CHKLIST.CPS. As has been noted before, early versions of *CPAV* will simply recreate the checksum file silently, thus covering the virus' path. Fortunately the version shipped as *Microsoft Anti-Virus* with DOS 6 uses a different file name, and from version 2 of the software this 'feature' has been removed.

Detection

Twitch presents the usual detection problems imposed by the use of a high-level language, with the added bonus of having large *Windows* libraries in the executable. The start of the code will be constant for all programs produced with the same version of the compiler, and all have to follow the same procedures required by a *Windows* program at initialisation.

The first step in detection is that the *Windows* entry point must be found, which involves analysing the much larger header of the NE file. Fortunately, no encryption or other camouflage is used, and offsets can be calculated to the sections of interest within the program. In fact, with this virus the entry point is always set to an offeset of 0A75h in segment 1 of the file.

You don't need to have *Windows* running to be able to find the virus, and because it cannot use the system to hide itself it could be reliably found by any scanner running within *Windows* too. However, it would still be a good idea to exit *Windows* on suspicion of the virus, to avoid it continuing to infect things while you're scanning. Twitch relies on *Windows*: if you exit the virus will be deactivated.

This virus again raises interesting questions about the security of scanning under *Windows*. The first *Windows* virus could simply infect files in the NE format. Twitch takes full advantage of many of the *Windows* features. Is anyone out there still relying on *Windows*-based detection?

TWITCH

Aliases: None Known.

Type: Parasitic, Memory-resident

Files: Windows Executables only.

Self-Recognition in Files:

Text string 'BEVWA' located at offset 2321h in infected files.

Self-Recognition in Memory:

None necessary - handled by internal Windows functionality.

Hex Pattern: The following pattern should be used with extreme care, as the virus is written in a high level language:

```
803e 2700 0074 0ebf 2700
1e57 ff36 7001 9a13 00ff
```

Intercepts: None - but hooks in to the Windows timer tick every 60 seconds.

Trigger: Causes Windows screen refresh (twitch) periodically.

Removal: Under clean system conditions, identify and replace infected files.

FEATURE

Memory-Resident Software: Pros and Cons

Wouldn't the world be a nicer place for the MIS Manager if computer viruses did not exist? As this is impossible - once discovered, something can hardly be uninvented - the next best thing iwould be an invention which would stop computer viruses in a completely trouble-free manner. Memory-resident anti-virus software seems to offer such a world... but does it live up to these claims?

The Different Options

There are three different types of memory-resident software: virus specific software, integrity checkers and behaviour blockers. Each of the different types has its own advantages and disadvantage, and these will be discussed later. However, all types of memory-resident software have one factor in common: they all attempt to monitor various aspects of the computer system and alert the user if something is amiss. In order to undertand the inherent strengths and weaknesses of this type of software, a brief discussion of how this works follows.

Excuse Me if I Interrupt

The simplest way to write memory-resident anti-virus software is to 'hook' the different interrupts which can be used for virus-like activity. At a basic level (which is sufficient for a general overview) these are Int 21h, the DOS Function Despatcher, and Int 13h, the BIOS Disk Services. This means that whenever a call is placed to either DOS or the BIOS, the anti-virus software can examine this call and decide whether it needs to take any action.

For example, a memory-resident virus scanner might intercept Int 21h function 4Bh, the DOS Load and Execute function, in order to scan executable files as they are loaded into memory. The execution of a program would then cause the following sequence of events:

When a user types the file name of the program which he wishes to execute, the DOS command interpreter identifies the command and executes the Int 21h instruction, with the appropriate registers set. This causes the processor to pass control to the code stored at the address pointed to by the relevant interrupt vector - which in this case points not to the DOS function despatcher but to the memory-resident anti-virus software. Before allowing the call to proceed, the anti-virus software will scan the file to ensure that the program is not infected before allowing the execute request to complete. In this way, the anti-virus software can monitor all standard calls to the system before a virus has the chance to execute, and infection is prevented, rather than merely detected.

Virus Scanners

Memory-resident virus scanners work on the above principle. Their behaviour is often modified so that they can detect viruses in files when those files are accessed. The majority of resident anti-virus software falls into this catagory.

There are several advantages to using this type of anti-virus software. The principal benefit is that it allows an organisation to take a pro-active stance towards virus detection - viruses are detected as they enter the system, rather than at some arbitrary time after infection. The other advantage is that the virus detection is carried out in the background without requiring any input from the user. Only in the event of the software finding a virus will the user be alerted to its presence.

A disadvantage of such virus-specific software is that it is very difficult to detect complex polymorphic viruses when constrained to tight memory limits - disk based virus scanners are not subject to the same constraint. Also, these programs require a large amount of system resources in terms of both processor time and memory (typically 20 to 60 K in size), which may prove unacceptable on older machines where resources are limited.

However, they provide good protection against the handful of common viruses which users are likely to encounter. Like all memory-resident software, it can be subverted, though out of all three classes, it is the least likely to be targetted (see below).

Integrity Checking Software

Memory-resident checksumming software works in exactly the same way as its disk-based bigger brother. Before any file is executed, its checksum is checked, and if it does not match the checksum in the database, the alarm is raised.

This memory-resident integrity software has the added benefit that users cannot load and execute applications which have not been approved. Certain packages, such as *McAfee* SCAN, only add checksum information to the database when the file has been scanned for viruses. This allows the IT Department to ensure that all executables brought onto company machines have been scanned for viruses.

The other advantage with intregity checkers is that they require less memory than virus scanners, as no large (and often expanding) virus database is needed. They are ideal for use in systems where memory is at a premium.

One drawback is that if the checksum is to be carried out over the whole file (as is required for maxium security) the time taken to load large files can become excessive. Also, the traditional benefits of integrity checking software (that is, the fact that it can detect new viruses) are eroded as when memory-resident, any new virus could target the software.

Behaviour Blockers

The last type of memory-resident software is known as behaviour blocking software. The actions of this software are just as its name suggests: it attempts to prevent certain functions being carried out by programs run on the machine.

A typical blocked function might be reformatting the hard drive, or opening an executable file for read write access. In addition to these services, many behaviour blocking products also prevent unauthorised TSR programs from becoming memory-resident.

Behaviour blocking can save the day when a computer virus triggers, and can also be a useful prophaliactic against infection, but is prone to be targeted. It adds the least processor load to the system, as it operates purely in a reactive manner. It should be noted that most memory-resident anti-virus products combine elements of all three types of protection in an attempt to thwart infectious code.

Know Thine Enemy

As memory-resident anti-virus software is widely used, some virus authors attempt to use stealth techniques in order to disguise the presence of their creations. In order to evade memory-resident software, there are a number of tricks available in the virus writers armoury.

The easiest way to evade detection by a behaviour blocker and integrity checker is to attempt to call the DOS Function Despatcher directly. The precise details of how this can be achieved will not be printed here (for obvious reasons), but suffice it to say that there are a number of different ways in which this can be achieved. The anti-virus vendor must make this process as difficult as possible in order for its product to be effective.

Conclusions

The most important point to note about memory-resident virus protection is that it can add extra security to a computer system *as long as it is not relied upon.* The danger is that both users and IT Managers will tend to forget that in itself, memory-resident software is by no means a guarantee of system integrity, but just as another line of defence

There are those in the industry who would argue that if a method cannot be relied upon then it should not be used. This attitude is at the other end of the scale: good memory-resident software *will* prevent accidents, such as rebooting a machine with an infected disk in the disk drive or copying Cascade-infected files onto the hard drive. Half a loaf is certainly better than none, and with these important caveats firmly in mind, there is much to reccommend the careful use of memory-resident anti-virus software.

COMPARATIVE REVIEW

TSR: Panacea or Snake Oil?

These results form the first comparative review of memory-resident anti-virus software ever undertaken by *Virus Bulletin.* The test protocol measures the simplest functionality which the TSRs claim to provide: detection of infected files as they are copied. The results may come as an unpleasant surprise for IT managers who have been *relying* on TSRs to protect machines in their domain.

McAfee Associates' VSHIELD

VSHIELD is the memory-resident offering from *McAfee Associates.* Not only does VSHIELD claim to be able to detect known viruses, but the developer has included the option to perform integrity checks on executable files.

Installation of VSHIELD proved to be simply a matter of copying the program onto the hard drive and adding a short line to the end of AUTOEXEC.BAT. For users wishing to protect workstations, a program named CHKSHLD is provided, which can be used to deny access to a network if the workstation is not running VSHIELD. This is done by reconfiguring the *Novell* Login Script, and the documentation provides a number of sample configuration files and examples of the correct command line options.

Like most of the other products reviewed, VSHIELD can be configured to use different amounts of memory. On the test machine, it took 41.6 Kbytes when loaded normally, 3.7 Kbytes when swapped to disk and 6.7 Kbytes when it was integrity checking only.

VSHIELD can be configured to provide four levels of protection. Level I is provided by the separate program VSHIELD1, which only checks the validation codes added by *McAfee Associate's* flagship product SCAN. Programs which fail the validation checks are not allowed to execute. Level II protection checks for known viruses at runtime, and a plethora of command line options exist to allow the user to configure this mode of operation. Level III protection provides a combination of the techniques applied in Levels I and II. Finally, Level IV protection allows the IT Manager to specify which programs can and cannot be executed.

The performance of SCAN was very good, gaining a creditable 78/83 against test-set one, and 242/250 against test-set two. The results of VSHIELD were not anything as encouraging, and rather set the tone for the other reviewed products. VSHIELD missed 15 viruses from test-set one (from these 15 viruses, SCAN identified 10 as infected) and a staggering 53 viruses from test-set two (of which SCAN could detect 45). The

PRODUCT	Non-resident Scanner (Test-set 1)	Memory-resident Component (Test-set 1)	Non-resident Scanner (Test-set 2)	Memory-resident Component (Test-set 2)
McAfee Associates' VSHIELD	94.0%	61.9%	90.6%	76.8%
Dr Solomon's TOOLKIT, GUARD	98.8%	95.2%	98.8%	97.2%
Microsoft Anti-Virus	80.7%	67.5%	64.8%	58.4%
Central Point Anti-Virus 2.0	89.2%	78.3%	73.6%	72.8%
Vi-Spy	96.4%	96.4%	100%	100%
F-Prot	98.4%	83.1%	99.6%	84.4%

Figure 1: Detection results for the virus scanner and memory-resident components within the virus packages. Only two products perform as advertsied, *F-Prot* and *Vi-sPY*. Note that with the exception of *Vi-Spy*, every memory-resident scanner performs worse than its main scanner in the product. Users should be warned of this in the documention.

documentation supplied with VSHIELD positively implies that it is capable of detecting all these viruses - clearly something is amiss.

These results beg the question of why so many samples were missed? Is it a case of poor quality control or does VSHIELD simply not look for these viruses? *McAfee Associates* was asked to comment on these results but has not done so - maybe it will feel more inclined to reply to its customers.

Dr Solomon's AVTK GUARD

VirusGuard forms part of *Dr Solomon's AVTK*, and is designed 'to monitor files for viruses and prevent infected programs from being run or copied'.

Installation of the program is easy, as it can done as part of the *Toolkit's* installation routine. The appropriate changes are made to AUTOEXEC.BAT, and once the machine is rebooted it is protected by GUARD. The documentation provided with the product is good, and gives a clear no-nonsense explanation of exactly what the program is intended to do, and which files and extensions are checked.

GUARD automatically attempts to load itself into enhanced memory, first trying XMS, and if this is unavailable, EMS. If neither XMS or EMS is available it defaults to accessing the virus signatures required from disk. These options all have command line overrides built in, so if there is a reason that the user wishes to use EMS instead of XMS the program can be configured accordingly. GUARD can be loaded either as a device driver or a TSR component.

VirusGuard protects against file viruses, both when infected files are executed or copied. When GUARD finds a virus, a red box is displayed on the screen, and the PC speaker emits a buzz. The message displayed when a virus is detected (and indeed most of the other features within GUARD) can be configured easily by using command line options.

The detection results of *FindVirus*, the scanner part of the *Toolkit*, performed very well, missing only 1 virus (Tremor) from test-set one, and 3 viruses out of the 250 in test-set 2. The results from GUARD were less impressive: four viruses slipped through from test-set 1 (Todor, Tremor, V2P6 and WinVir14) of which *FindVirus* was capable of detecting all except for Tremor. The same story was true of test-set 2, where GUARD missed 7 infected files, of which *FindVirus* could detect 4.

These result show that GUARD is incapable of detecting certain polymorphic viruses, and *S&S* was asked to comment on this. Note that the documentation describing *VirusGuard* states that 'When *VirusGuard* is installed, it will sit in the background, checking for **all known viruses**...' *[VB's emphasis. Ed.]*

Iolo Davidson from *S&S* explained that there *were* a few exceptions to this rule. The reason for these omissions was that 'the routines take up a lot of memory-resident space'. Dr Solomon later added that he was 'flabbergasted' that the manual did not explain this clearly, and that this situation would be rectified. Such changes to the manual would be a welcome addition to GUARD, which is basically a good utility. However, Davidson warned that it should not be relied on, as 'it's not the be-all and end-all of virus detection - it's really there just to prevent accidents.'

Microsoft's MSAV

The anti-virus component supplied with *MS-DOS 6* is a subset of *Central Point's* popular anti-virus product *CPAV*. Just like *CPAV*, *MSAV* contains its own memory-resident scanner, *VSafe*.

According to the documentation (of which a scant 17 pages describes all the components of *MSAV*), *VSafe* is 'a memory-resident program that monitors your computer and warns of changes that might have been caused by a virus.' This is never really explained further in the manual, but from examining the options within *VSafe*, it appears to be a combined behaviour blocker and memory-resident scanner.

VSafe can either be configured from the command line or, once memory-resident, by popping up a configuration box using a 'hot-key'. This allows the product to be operated easily and simply.

Once memory-resident, *VSafe* takes up 23K of conventional memory and 23K of XMS, making it one of the more memory-hungry programs reviewed.

The performance of the *MSAV* package was disappointing. The scanner was only capable of detecting 67 of the 83 viruses in test-set number 1 - this went down to 162 out of 250 in the second test-set. However, the results obtained by *VSafe* provided more cause for concern. From the first test-set, 27 files were missed (of which *MSAV* could detect 11), and against the second, 104 viruses slipped through, 28 of which were detectable by *MSAV*. A fairly dismal result.

Central Point's CPAV 2.0

CPAV 2.0's memory-resident scanner, *VSafe*, claims to be 'a comprehensive memory-resident, virus protection utility, that provides real time monitoring.' The product is easily configured and offers such added extras as an audit trail and the ability to communicate with its big brother, *CPAV* for *NetWare*, allowing centralised virus protection.

The documentation supplied with *VSafe* is clear and easy to read, but is unfortunately let down rather by its index. Its style is rather irritating to the seasoned anti-virus reviewer, but for users who will lack specialist knowledge on the subject, the over-simplified way in which the problems are tackled will almost certainly be a benefit. The installation instructions are easy to follow, and allows users to configure the system to suit the system they are using.

The memory-resident part of *CPAV* comes in two different guises. *VSafe*, which can be loaded either as an EXE file or a device driver and *VWatch*. The difference between the two programs is that *VWatch* is a virus-specific program, whereas *VSafe* can also use the checksum information generated by *CPAV* to check for changes to executable files.

In terms of detection, *CPAV 2.0* (using the latest available signature update files) did little better than *MSAV*. The scanner detected 74 of the 83 viruses from test-set one, and a disappointing 184 out of 250 from test-set two.

As has been the case with all the other products tested, the memory-resident utility *VSafe* missed viruses detected by the scanner. In this case *VSafe* missed 18 viruses from test-set one, nine of which were detected by *CPAV*, and 68 from test-set two. Only seven of the test-set two viruses missed by *VSafe* were detected by *CPAV*, but five viruses detected by *VSafe* were not detected by *CPAV*. *Central Point* explains these puzzling results by the fact that the TSR component uses a slightly different method of searching for viruses.

This niggle aside, *VSafe* is clearly not detecting all of the same viruses as *CPAV*, though *Central Point* claims that any viruses missed by the virus-specific part of *VSafe* will be detected by the generic virus detection included. Again, this policy is not made clear anywhere in the manual, and this omission should be rectified.

RG Software's Vi-Spy

Vi-Spy is the anti-virus offering from the American company, *RG Software*. It comprises of a virus scanner and integrity checker, and a memory-resident scanner, RVS, in addition to several utilities to help clean up a virus attack.

RVS makes some pretty tall claims as to its functionality: IOt was stated that RVS would detect viruses no matter how they were written to the disk, including COPY, XCOPY and from within Windows. Pretty tall claims: could RVS live up to them?

The documentation provided with *Vi-Spy* is brief and to the point, consisting of a 'Computer Virus Primer and Trouble-shooting Guide' and a short 45-page user manual. The action of RVS is explained in a non-technical manner, and then a list of command line options is provided so that the user can configure the system to suit his needs.

Installation of RVS was easy: the installation program took care of the changes to the AUTOEXEC.BAT file, after asking the user if he wished to install the memory-resident software.

RVS can be run in several different modes. If upper memory is available, it takes just 19K of upper memory and 60K of Expanded memory. If no expanded memory is available, the user has the choice of using 79K of memory or swapping parts of RVS to disk, saving 60K, although the penalty for this is a loss of performance.

The test results for *Vi-Spy* and RVS are by far the best in this comparative review: in fact, RVS was the only TSR virus protection which detected all the viruses detected by the main scanner included with the package.

When tested against test-set one, both RVS and *Vi-Spy* missed three infected files, one sample of the Loren virus and two infections of Necros. The results against test-set two were even more impressive: both RVS and *Vi-Spy* gained a perfect score of 250/250. These results show that it is possible to write a memory-resident scanner which is capable of detecting the more complex polymorphic viruses. Thoroughly recommended.

FirskSoftware'sF-Prot

The memory-resident portion of *F-Prot*, VIRSTOP, is the only product on test which makes it clear in its documentation that it makes no attempt to detect polymorphic viruses. In paragraph four of the VIRSTOP documentation, the warning is made quite clear: 'IMPORTANT! ... VIRSTOP does not detect the same number of viruses as F-PROT. In particular, VIRSTOP does not detect most polymorphic viruses. It is therefore recommended that VIRSTOP only be used as one component of the virus protection - do not rely on it alone.'

The remainder of the documentation goes on to describe the different command line switches which VIRSTOP recongises. These are very simple, and allow the user to configure the behaviour of the software (such as which files are scanned and when) without uneccessary complexity.

When memory-resident, VIRSTOP takes up 16Kbytes of memory, although it can be loaded high, either as a device driver or from AUTOEXEC.BAT. Like all the other products, if memory is a particularly scarce commodity, its virus signatures can be swapped out to disk, saving 12.4 Kbytes. Again, this leads to a large drop in performance.

The performance of VIRSTOP was slightly disappointing (although better than *McAfee Associates*, who did not even include a caveat about polymorphic viruses). The scanning part of *F-Prot* missed three viruses from test-set one (an EXEBUG 2 dropper and two infections of the Loren virus), and gained a near perfect 249/250 and one 'suspicious' file when run against test-set two.

VIRSTOP, on the other hand, missed 14 files from test-set one (of which F-Prot detected 11) and 39 files from test-set two (of which *F-Prot* detected 38 as infected and one as suspicious). Given that the user is warned from the outset of what the software is designed to accomplish, this is fair.

System Load

The performance overheads added by the different products varied widely. In order to gain some measure of the impact a user could expect on his system, the time taken to copy 49 clean executable files from a floppy diskette to the hard drive was measured with each of the memory-resident programs monitoring copied files. The results obtained are shown in the following table.

PRODUCT	TIME
CPAV	5:04
MSAV	5:08
AVTK	2:19
VI-SPY	2:08
F-PROT:	1:51
VSHIELD:	3:56
None:	1:33

The overhead added by *MSAV*, combined with its poor detection results, make its results completely unacceptable, with *CPAV 2.0* and *McAfee* faring little better. The load added by *Dr Solomon's AVTK* and *Vi-Spy* is far more reasonable, but it could become rather restrictive. Note that not only was *Vi-Spy* the most accurate memory-resident program, but it was the second fastest. Surely its competitors have something to learn from these results?

Conclusions

The poor performance of all the memory-resident scanners came as something of a shock when conducting this review. With the exception of *Vi-Spy*, in every single case, the memory-resident scanners were incapable of detecting viruses which were clearly known to the product developers, most of whom did not point this out in their manuals.

Vi-Spy is the only product to come through this test unscathed, and is thoroughly recommended. As for the other products, it seeems that the inference of 'detects all viruses' has slipped in accidentally to the manufacturers' documentation - it is to be hoped that these test results will be reminder enough for this to be corrected.

Vendor Details

McAfee Associates.
Tel. +1 (408) 988 3832. Fax. +1 (408) 970 9727
S&S International Ltd.
Tel. +44 (442) 877 877. Fax. +44 (442) 877883
Microsoft Ltd.
Local support varies.
Central Point Software.
Tel. +1 (503) 690 8080. Fax. +1 (503) 690 7133
RG Software.
Tel. +1 (602) 423 8000. Fax. +1 (602) 423 8389
Frisk Software.
Tel. +354 (1) 617273. Fax. +354 (1) 617274

Technical Details

Tests were carried out on an *Opus Technologies* 386SX 25Mhz, with 4MB RAM, a high-density 3.5-inch drive, a high-density 5.25-inch drive and an 80 Mbyte hard drive.Timing tests were taken by measuring the time taken to copy 49 files (comprising 1.37 Mbytes) from the floppy drive to the hard drive.

The viruses used in the test-sets were all genuine infections of parasitic file viruses. Test-set one was drawn from a subset of the *Virus Bulletin* 'In the wild' test-set. Test-set two was made up of 250 different file infecting viruses drawn randomly from the *Virus Bulletin* collection.

PRODUCT REVIEW

ThunderBYTE is GO!

Keith Jackson

ThunderBYTE has been reviewed before by *VB*, but in the far-off days of August 1991 it comprised a plug-in PC card and associated software. This review looks at the software component only. The *ThunderBYTE Anti-Virus* package comprises several utilities which, amongst other things, will detect viruses using scanning, checksums and 'heuristic' techniques [*'Heuristic analysis' is techno-speak for an educated guess. Ed.*], clean viruses from infected files, and provide immunisation against virus infection.

Memory-resident programs are provided which offer scanning, integrity checking, and guard programs which prevent assorted unwanted events. Individual components can either execute as stand-alone DOS programs, or be activated from within an all-encompassing shell program which offers drop-down menus and simple selection of the various facilities which are on offer. All in all, *ThunderBYTE* seems to be quite a comprehensive anti-virus package.

Installation

Installation of *ThunderBYTE* proved to be trivial. The name of the subdirectory to which files are to be copied must be stated and, optionally, the name used for the *ThunderBYTE* reference files (see below) can be specified. The user is prompted as to whether each of the memory-resident utilities should be installed, and a batch file is created which sets up the chosen utilities.

Suitable amendments are made to the start-up files to ensure that this batch file is executed every time that the PC is booted, and the original AUTOEXEC.BAT file is saved as a backup file. The installation program then scans the hard disk, and creates a reference file in every subdirectory which contains executable files.

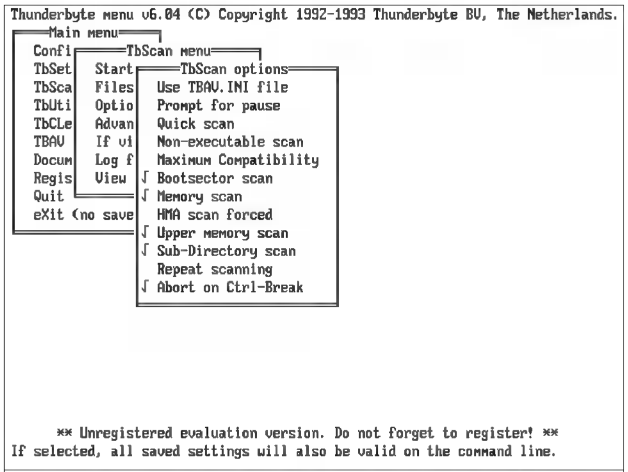
All this proved very simple to do, the only point of note being a message which appeared warning me that I was using the shareware version, and consequently the installation process may differ from what is normally used. The software was provided on a 3.5 inch floppy disk, I do not know how those users who only have a 5.25 inch disk drive are catered for (if at all).

Documentation

The documentation is voluminous, helpful and well organised, though the omission of an index mars its usability. All operations carried out by *ThunderBYTE* are well explained, and complete sections are included describing general techniques such as anti-virus strategy, what to do if a virus is detected, and how to use the various utilities.

The documentation makes it clear that much development effort has been expended on making *ThunderBYTE* very fast, and the programmers state their belief that the addition of a pretty front-end such as a *Windows* GUI (Graphical User Interface) 'inflates program size, performs more sluggishly and puts a penalty on overall reliability'. I could not agree more, and commend *ESaSS* for ignoring the trend toward making anti-virus software into beautifully sculpted *Windows* programs. The best anti-virus programs are those that require hardly any memory, impose almost zero overhead, and remain invisible, only revealing their presence when they detect something untoward.

During installation, *ThunderBYTE* creates a reference file in each subdirectory for checking the fule integrity in the future. These files are updated as necessary when any 'permitted' changes to executable files are made. The logic behind this decision is that it is possible to add new software onto the machine simply by adding a new directory which includes a TBAV checksum file. This may be useful in a large company but on a stand-alone machine is a pain in the neck - surely this could be an option.



```
Thunderbyte menu v6.04 (C) Copyright 1992-1993 Thunderbyte BV, The Netherlands.
╔═══Main menu═════╗
  Confi╔═════TbScan menu══════╗
  TbSet   Start╔════TbScan options═══════╗
  TbSca   Files    Use TBAV.INI file
  TbUti   Optio    Prompt for pause
  TbCLe   Advan    Quick scan
  TBAV    If vi    Non-executable scan
  Docum   Log f    Maximum Compatibility
  Regis   View   √ Bootsector scan
  Quit           √ Memory scan
  eXit (no save    HMA scan forced
                 √ Upper memory scan
                 √ Sub-Directory scan
                   Repeat scanning
                 √ Abort on Ctrl-Break

     ** Unregistered evaluation version. Do not forget to register! **
If selected, all saved settings will also be valid on the command line.
```

The front end to the *ThunderBYTE* software is excellent allowing users quickly and simply to configure the myriad options offered.

Probably the Fastest Scanner In the World...

Using the default setup, *ThunderBYTE* checked the hard disk on my test computer (which contains 693 files occupying 21 Mbytes spread across 23 subdirectories) in just 19 seconds. Amazingly enough, this was not even in quickscan mode, where it took just 17 seconds. These figures may not sound fast in absolute terms, but my test computer is only a 16 MHz 386, not a particularly fast PC, and to put *ThunderBYTE*'s time into perspective, scanning the same hard disk took *Dr Solomon's Anti-Virus Toolkit* 53 seconds, and *Sweep* from *Sophos* required 6 minutes 25 seconds in normal mode, and 1 minute 53 seconds in Quick mode.

These tests show quite clearly that *ThunderBYTE* is *seriously* fast, and that the claim made in the documentation that *ThunderBYTE* is 'the fastest scanner on the market today', is probably true and not just the marketing-speak common with most anti-virus products. When one realises that *ThunderBYTE* is carrying out both a scan and an integrity test (checksum verification) in the above check the above quoted times seem even more impressive.

The speed at which disks are checked seems to be maintained by using one of several algorithms when inspecting file(s). These algorithms are described within the documentation, and comprise different methods of trying to detect the presence or absence of a virus. They are used unless the structure of the file under study prevents their application, in which case *ThunderBYTE* resorts to the more traditional tactic of scanning each byte for a virus signature.

There is not enough space within this review to provide complete details of the five algorithms described in the documentation. While *ThunderBYTE* is scanning a disk it provides very detailed information about each file that it checks on screen. On all but the smallest disks, this whizzes by so quickly that it cannot be read, but this is not a problem as the logging facilities allow all or partial details of every scan to be captured to file.

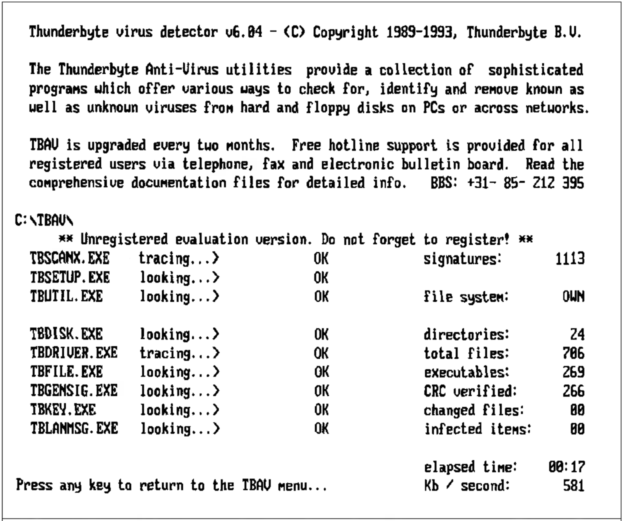
The Family Tree

The documentation states that most of the signatures used by *ThunderBYTE* are 'family signatures', that is, a single signature detects several viruses. This concurs with my tests where many viruses were detected as 'Jerusalem.something', or 'Jerusalem like'. This is introducing by default a hierarchical naming convention similar in structure to that used in taxonomy for the naming of animal species. Do users really care that their computer has Jerusalem.Blancmange? I think not, and this naming convention seems quite reasonable.

"the claim made ... that ThunderBYTE is 'the fastest scanner on the market today', is probably true."

All of the various properties which can be detected heuristically by *ThunderBYTE* are explained in an Appendix to the documentation, and from this long list I detected during my tests warnings about files that were (among other things) compressed, contained 'suspicious' jump instructions, or contained garbage instructions.

The most intriguing warning stated that a particular file (a *Windows* system executable) 'contains code that seems to have no purpose other than encryption or avoiding recognition by virus scanners'. There has recently been much discussion in the computer press about *Microsoft* including encrypted code within *Windows 3.1* whose function appears to be to refuse to



```
  Thunderbyte virus detector v6.04 - (C) Copyright 1989-1993, Thunderbyte B.V.

  The Thunderbyte Anti-Virus utilities  provide a collection of  sophisticated
  programs which offer various ways to check for, identify and remove known as
  well as unknown viruses from hard and floppy disks on PCs or across networks.

  TBAV is upgraded every two months.  Free hotline support is provided for all
  registered users via telephone, fax and electronic bulletin board.  Read the
  comprehensive documentation files for detailed info.   BBS: +31- 85- 212 395

C:\TBAV\
     ** Unregistered evaluation version. Do not forget to register! **
  TBSCANX.EXE    tracing...>           OK            signatures:       1113
  TBSETUP.EXE    looking...>           OK
  TBUTIL.EXE     looking...>           OK            file system:       OWN

  TBDISK.EXE     looking...>           OK            directories:        24
  TBDRIVER.EXE   tracing...>           OK            total files:       706
  TBFILE.EXE     looking...>           OK            executables:       269
  TBGENSIG.EXE   looking...>           OK            CRC verified:      266
  TBKEY.EXE      looking...>           OK            changed files:      00
  TBLANMSG.EXE   looking...>           OK            infected items:     00

                                                     elapsed time:    00:17
Press any key to return to the TBAV menu...          Kb / second:       581
```

ThunderBYTE is probably the fastest scanner ever reviewed by *Virus Bulletin*, and offers the potential user a great deal.

let *Windows* operate with any version of DOS other than *Microsoft's* own, so *ThunderBYTE* may well be correct in its judgement.

How *ThunderBYTE* does all this I have no idea. The manual is clear about how the heuristic tests are used, but for obvious commercial considerations it does not spell out the exact details. In particular I would be very intrigued to know what tests are used to detect 'garbage' code. Even though the warnings listed above were flagged in the log file, they did not cause *ThunderBYTE* to think that any of the files were virus infected. This is probably because whilst scanning, it has the sense to exclude certain files from heuristic analysis; such files are however scanned in the normal way. This tactic seems to have paid off, as *ThunderBYTE* did not produce a single false positive detection during all of my testing, even when the heuristic detection was wound up to its highest level. A creditable performance.

Blistering Speed, Reasonable Accuracy

Given that *ThunderBYTE* can check disks very quickly, does this affect its virus detection performance? It failed to detect just 33 of the 228 viruses in my test-set (see *Technical Details* section below). This corresponds to a detection rate of 87%, which is reasonable but not as good as the best scanners around. ESaSS claims that these detection results are poor because the heuristics treats files without an executable extension differently, and if the files were renamed to an executable extension the problem would be resolved - if this is the case, it should be clearly stated in the manual.

One rather curious result was that the viruses which were not detected were mainly ones that have been known about for some time (a few years). All of the 13 viruses added to my test-set a few months ago were detected correctly, and only 3 out of 30 viruses added to the test-set a year ago were not detected. This is most curious as it is usually the newer viruses that are not detected. Either the amount of effort put into signature extraction has been increased recently by the developers, or the use of 'families' of signatures described above has started to pay off, and detection of variations on a theme has become easier. Only time will tell. Either way, *ThunderBYTE* is updated every 2 months, so it should readily keep up with new viruses.

Memory-resident Protection

The amount of memory which is required by the memory-resident utilities is explained in detail in the documentation, along with a few hints as to how memory usage can be reduced. I measured the memory used, and with the exception of the memory-resident scanner (which requires 25 Kbytes), they are all very small, ranging from 800 bytes up to 1.4 Kbytes.

I chose to let *ThunderBYTE* install four utilities which provided memory-resident scanning, integrity checking, file guard and memory guard facilities. I had all of these active whilst carrying out the testing for this review, and can confirm that with the exception of minor problems with regard to the loading of *Sidekick* (see below), they do not interfere with the actual operation of the PC in any noticeable way.

Suffice it to say that *ThunderBYTE*'s memory-resident utilities are the only ones that I have seen in a while that I might actually recommend using. I do however assume that the annoying shareware banner that nags the user to register is removed (how to deter potential customers at a stroke!), and that the user has actually requested memory-resident anti-virus utilities in the first place.

ThunderBYTE seems very successful at detecting when a program not listed in its exception list is attempting to become memory-resident. However I did find one foible whilst testing this process. When *Sidekick* attempts to become resident, this is detected, and the user is asked if this should be allowed or not. If this attempt is disallowed, and then another attempt is made to make *Sidekick* memory-resident, then *Sidekick* itself refuses the second time, saying that it already is memory-resident and cannot be loaded for a second time! Something about *ThunderBYTE's* refusal is making *Sidekick* very confused.

Once a user has stated that a particular program can become memory-resident, then *ThunderBYTE* alters its reference file(s) and permits all future memory-residence attempts.

Does *ThunderBYTE* mind its own reference files being changed? My tests showed that if the first few bytes of a reference file were altered, then this is not noticed. However wholesale changes of data *are* noticed and the integrity checker rightfully complains about this. Does a change to what looks like it is probably a header sequence actually matter? Probably not, but the developers would do well to either prevent *any* changes from being accepted, or to document the present reference file structure - and if the header information is not used, why is it there?

Bugs and Gripes

The previous *VB* review of *ThunderBYTE* found some problems with the test computer locking up and displaying 'Fatal Error' messages. These all appear to have been cured, and I saw nothing untoward displayed by the memory-resident parts of *ThunderBYTE* while reviewing them.

In fact, the only problem I found which was worth commenting on was that some of the menu options offered by the *ThunderBYTE* shell program do not always seem to remain set. Quite often I selected a drop-down menu, activated an option such as creating a log or inspecting non-executable files, ran a scan, and then found that the results had not followed my selected choice. On inspecting the drop-down menu again the option was then mysteriously not set. Even after much experimentation I never got to the bottom of this, and I could not establish a consistent cause and effect pattern. The developers would be advised to look into this.

Conclusions

The most impressive thing about carrying out this review was without a doubt the blinding speed at which *ThunderBYTE* can scan a disk. This is not a comparative review so I cannot exhaustively validate the claim made by the developers that *ThunderBYTE* is the fastest scanner on the market, but my tests show that if it is not the fastest, it is certainly up there with the quickest. *ThunderBYTE* by name, thunder byte by nature?

This speed is coupled with an adequate detection capability, more features than I could possibly test, and several resident utilities. *VB* reviewed *ThunderBYTE* favourably last time around, with some reservations. I like it even more this time around , with no reservations beyond those mentioned in this review. Recommended.

Technical Details

Product: *ThunderBYTE*

Developer: *ESaSS B.V.*, Kerkenbos 10-21,6546 BB Nijmegen (also P.O.Box 1380, 6501 BJ Nijmegen), The Netherlands, Tel: +31 (0) 80 787881, Fax: +31 (0) 80 789186, Support BBS: +31 (85) 212 395, FidoNet: 2:280/200, email: veldman@esass.iaf.nl

Availability: Any *IBM* compatible PC with at least 1 Mbyte of free disk space. *ThunderBYTE* will operate with any version of DOS from 3.0 upwards, though version 5.0 or later is recommended.

Version evaluated: 6.04

Serial number: None visible

Price: $115 with updates via BBS. Update service is $180 extra.

Hardware used: *Toshiba 3100SX*, a 16MHz 386 laptop, with 5 Mbytes of RAM, one 3.5 inch (1.44M) floppy disk drive, and a 40 Mbyte hard disk, running under MS-DOS v5.0.

Viruses used for testing purposes: This suite of 143 unique viruses (according to the virus naming convention employed by VB), spread across 228 individual virus samples, is the current standard test set. A specific test is normally made against 1024 viruses generated by the Mutation Engine (which are difficult to detect with certainty), but was not performed in this review *[Keith, you are fired! Ed.]*.

Full details of the test-set used are printed in *Virus Bulletin*, August 1993, p.19.

TUTORIAL

Putting The Boot In

The prerequisite for reliable use of anti-virus software is a cleanly booted system. The last time one wishes to be left with no secure method of booting a machine is when there is known to be a virus ‘doing the rounds’, and a proactive step towards protecting one's data is to create a boot disk before disaster strikes. The best catastrophes are the ones which can be prevented!

On an unadorned MS-DOS system, it is very easy to create a bootable diskette - it is just a matter of formatting a disk with the /s option selected. However, life is more complicated when proprietary disk compression software, such as *SuperStor* or *Stacker* is used. In this case, inserting a bootable DOS system disk *will* boot the machine, but all the files stored on the compressed part of the drive will be inaccessible. Here we examine the most popular compression systems, and how to create a boot disk with them.

Quarts into Pint Pots...

In order to understand why one needs a special boot disk, it is worth considering how disk compression software works. The following discussion describes a general disk compression utility, and not all the details will fit all three products examined here. However it should be sufficient to gain a basic understanding of the relevant issues.

The majority of data compressions algorithms rely on ‘patterns’ which occur within the data. For example, a text file contains a great deal of structure, and most products can take this structure into account when compressing files. The way the drive is compressed is analogous to what happens when one PKZIPs or DIETs a file.

In the case of disk compression, the compression and decompression needs to be done in real time - that is, as a file is accessed it is decompressed automatically. This is achieved by having the disk compression software in memory, either by including it in the operating system, or loading it as a device driver.

The software then monitors hard disk accesses and alters them so that the information requested is automatically decompressed before being returned to the calling function. Before the data stored on the compressed partition on the hard drive can be accessed, the disk compression software must be memory-resident.

MS-DOS 6.0

The easiest compression system for which to create a bootable system disk is *Doublespace*. As *Microsoft* has complete control of the standard which defines the operating system it has included an extra hidden file on all DOS 6 bootable diskettes called DBLSPACE.BIN. This file is loaded automatically at boot time and the code which it contains handles all the calls to the hard drive.

Therefore in order to create a bootable disk using DOS 6, one simply uses the format command, ‘FORMAT A: /S’.

SuperStor

AddStor, the maker of *SuperStor*, did not have the luxury of being able to rewrite the operating system in order to facilitate inclusion of its compression software, and has therefore had to go along the route of creating device drivers which can intercept calls to the hard drive. Creating a boot disk for the *SuperStor* system is therefore more complicated, but can be done as follows:

Create a bootable floppy disk using the command FORMAT A: /S. Next, copy the files SSTORDRV.SYS and DEVSWAP.COM onto the disk. The CONFIG.SYS file on the floppy disk drive should read as follows:

```
DEVICE=A:\SSTORDRV.SYS
DEVICE=A:\DEVSWAP.COM  FILES=20  BUFFERS=20
```

Stacker

Stacker works by loading a device driver which can ‘mount’ the stacked drive. Due to the way in which the device driver is called, creating a boot disk is slightly more complicated. However, by following the instructions given below a working boot disk should be created:

1. Format a bootable DOS system disk by using the command FORMAT a: /s
2. Copy the file c:\stacker\stacker.com to the floppy disk.
3. Copy the file c:\stacker\sswap.com to the floppy disk.
4. The file c:\config.sys should have two lines in it which look something like:

```
DEVICE=C:\stacker\stacker.com C:\stacvol.dsk
DEVICE=C:\stacker\sswap.com C:\stacvol.dsk /sync
```

These lines should be copied into a file on the floppy disk. However, the entry ‘c:\stacker\stacker.com’ should be altered to ‘a:\stacker.com’, and the entry ‘c:\stacker\sswap.com’ should be altered to read ‘a:\sswap.com’. This is vital as otherwise the system will load files from the hard drive. It is important not to alter any other parts of these two lines.

It can be seen that for very little effort is is possible to create a clean boot disk for a compression system. Time spent doing this now will repay itself many times over in the event of a virus outbreak.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues) including first-class/airmail delivery:

UK £195, Europe £225, International £245 (US$395)

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, 21 The Quadrant, Abingdon Science Park, Abingdon, OX14 3YS, England

Tel. (0235) 555139, International Tel (+44) 235 555139
Fax. (0235) 559935, International Fax (+44) 235 559935

US subscriptions only:

June Jordan, *Virus Bulletin*, 590 Danbury Road, Ridgefield, CT 06877, USA

Tel 203 431 8720, Fax 203 431 8165

END NOTES AND NEWS

Copies of the proceedings from the *1993 Virus Bulletin Conference* are available from *Virus Bulletin*, priced at £50.00 plus p+p. For further information contact Victoria Lammer. Tel. +44 (235) 555139.

Central Point has announced a **reduction of the price** of *Central Point Anti-Virus 2.0.* The package now costs £99, and includes four updates Commenting on the price drop, Jim Horsburgh, Managing Director of *Central Point Software*, said 'As the number of viruses continues to grow, users need to know that they have the most up-to-date virus protection available.' Tel. +44 (81) 848 1414.

Next to jump onto the 'competitive upgrade' bandwagon is *McAfee Associates*, which has announced that **it will sell site licenses for its products at half price** to new customers who can show proof of ownership of competing anti-virus products. Tel +1 (408) 988 3832.

The *20th Annual Computer Security Conference and National Exhibition* will be held in **Anaheim, California on November 8th-10th**. The conference, sponsored by the *Computer Security Institute*, attempts to address the full range of computer-security issues faced by IT professionals. According to conference chairman, William H. Murray of *Deloitte & Touche*, 'this conference is designed by practitioners for practitioners to meet the changing needs of information security'. For further information, contact Patrice Rapalus. Tel +1 (415) 905 2310.

The habit of leaving text messages inside executable code is not exclusively reserved for virus writers. 'Eli and Yuval were here' reads the legend in a certain virus scanner's *Windows* DLL. Maybe that is why *Windows* executables are so large...

The *16th National Computer Security Conference* , sponsored by the *National Institute for Standards and Technology* and the *National Computer Security Center*, will be held at the Baltimore Convention Center, USA, on September 20th-23rd. Tel. +1 (301) 975 3872.

Leprechaun Software has announced the launch of its new NLM, *Network Security Organiser*. **The product is capable of analysing the network configuration's vulnerability to viruses**, and allows centralised updating of any anti-virus software. Charles Rutstein, in an *NCSA* product evaluation, concluded 'For a five-user price of $250, no network should be without a copy.' High praise indeed. Tel. +1 (404) 971 8900.

According to a report in *The Nikkei Weekly*, **a record number of computer virus incidents were reported in Japan in May** . This figure is 2.3 times higher than the corresponding period last year.

Moves are afoot to strengthen the laws governing computer crime in Singapore. The proposed *Computer Misuse Bill* (which is modelled on the UK *Computer Misuse Act*), will cover four main areas: Hacking, Unauthorised Modifications, Theft of Computer time and 'misuse'.

S&S International has announced a new guaranteed turnaround time for its data recovery service. The company claims that in cases where they take longer than 48 hours between receiving the disk and completing the recovery, *S&S* will discount its published prices by 20%.

Virus Alert: three new viruses are known to be in the wild in the United Kingdom. The first, **Globe**, is a simple file infector written in Turbo Pascal. The virus was discovered due to the vigilance of a Network manager, who isolated a sample of the virus. It is believed to be of Russian origin. The other two viruses are both variants of the Shoo virus. The virus plays a tune when triggered. None of these viruses is believed to be widespread at this time.

Digital Equipment Corporation has signed a **US$250,000 software contract** with *Sophos Ltd*. As part of the contract, *Digital* has acquired a worldwide licence to use *VSweep*, *Sophos*' anti-virus package for *VAX* and *Alpha AXP* servers running *Pathworks*. *Digital* will also sell the product through its channels worldwide. Tel. +44 (235) 559933.